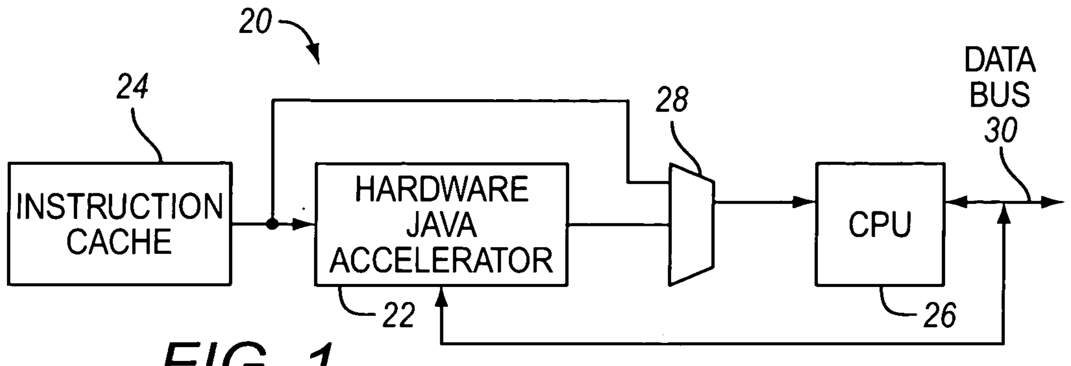

20
24
INSTRUCTION CACHE
HARDWARE JAVA ACCELERATOR
22
28
CPU
26
DATA BUS
30

FIG. 1

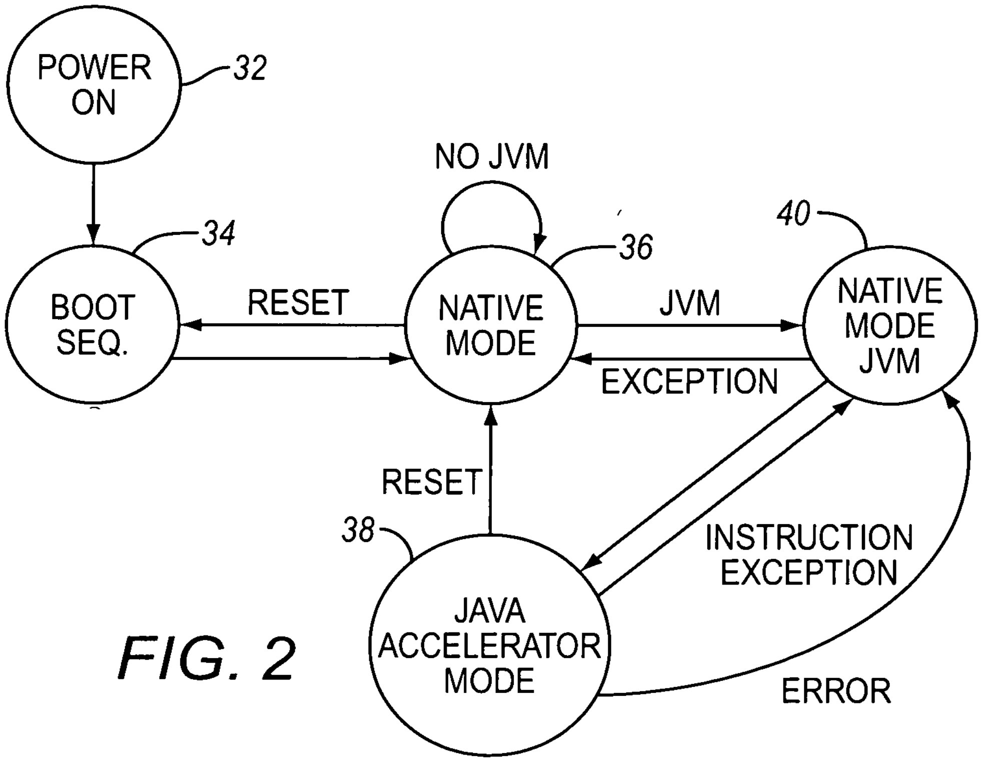

POWER ON
32
BOOT SEQ.
34
RESET
NO JVM
NATIVE MODE
36
JVM
EXCEPTION
NATIVE MODE JVM
40
RESET
38
JAVA ACCELERATOR MODE
INSTRUCTION EXCEPTION
ERROR

FIG. 2

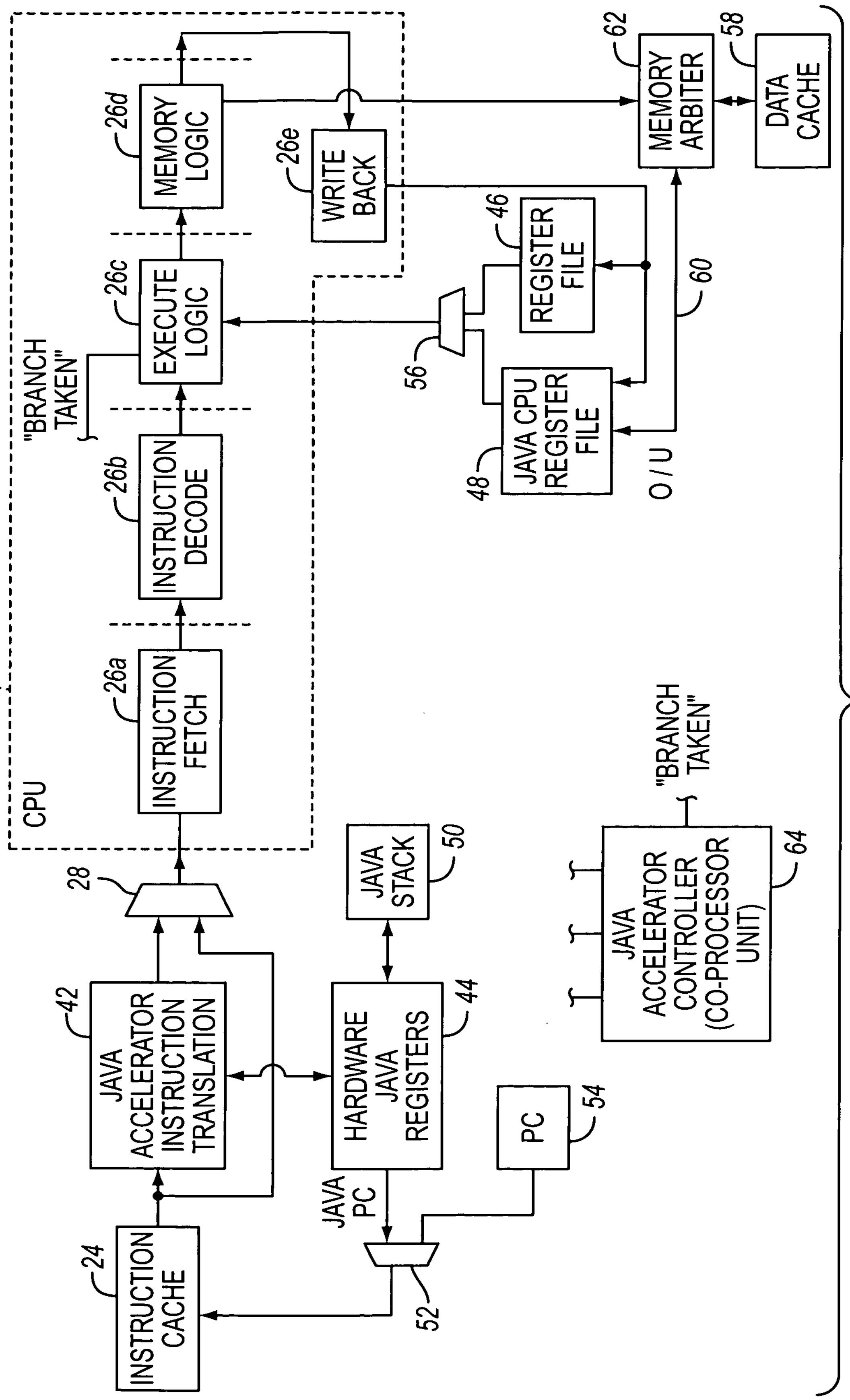

CPU
26
INSTRUCTION FETCH
26a
INSTRUCTION DECODE
26b
EXECUTE LOGIC
26c
"BRANCH TAKEN"
MEMORY LOGIC
26d
WRITE BACK
26e
MEMORY ARBITER
62
DATA CACHE
58
REGISTER FILE
46
JAVA CPU REGISTER FILE
48
56
60
O / U
28
JAVA ACCELERATOR INSTRUCTION TRANSLATION
42
HARDWARE JAVA REGISTERS
44
JAVA STACK
50
JAVA PC
PC
54
52
INSTRUCTION CACHE
24
JAVA ACCELERATOR CONTROLLER (CO-PROCESSOR UNIT)
64
"BRANCH TAKEN"

FIG. 3

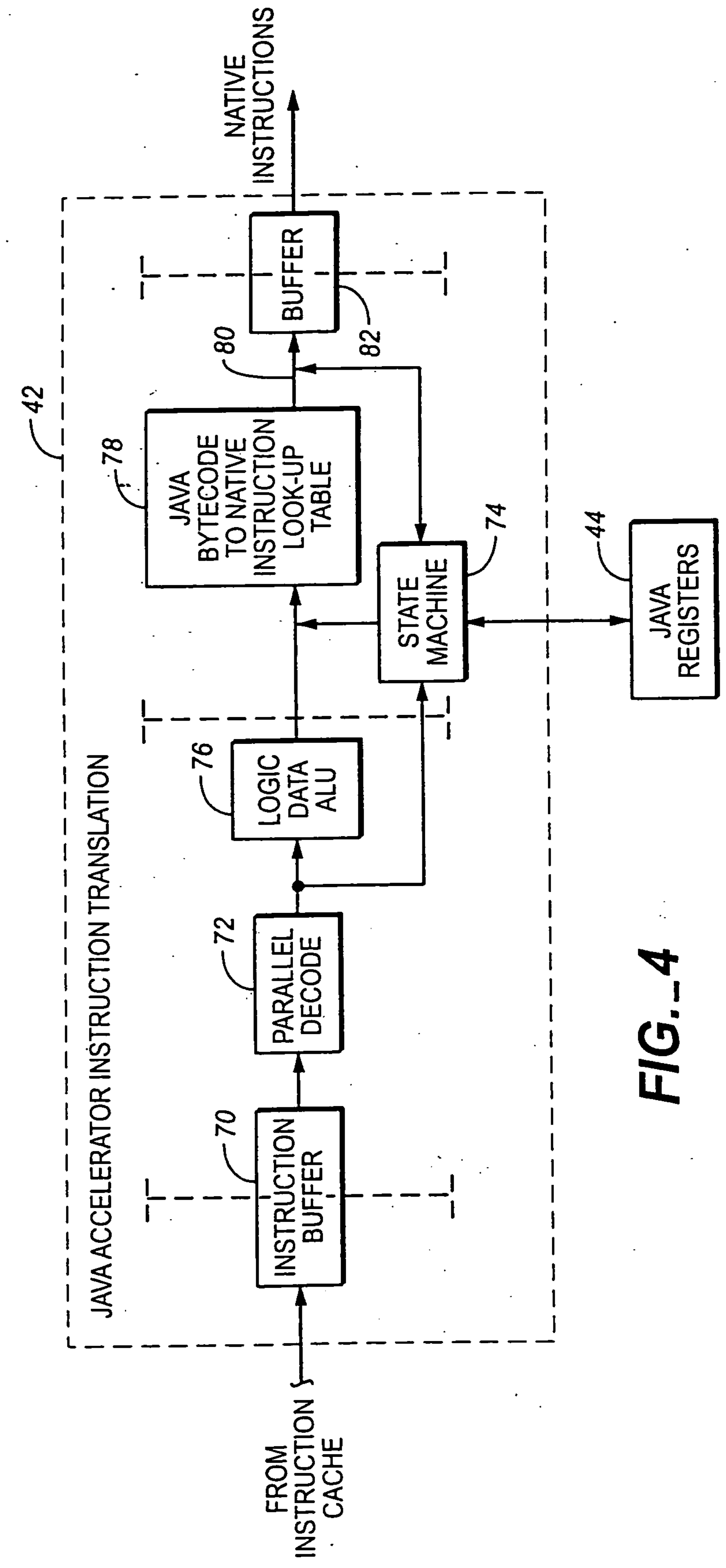
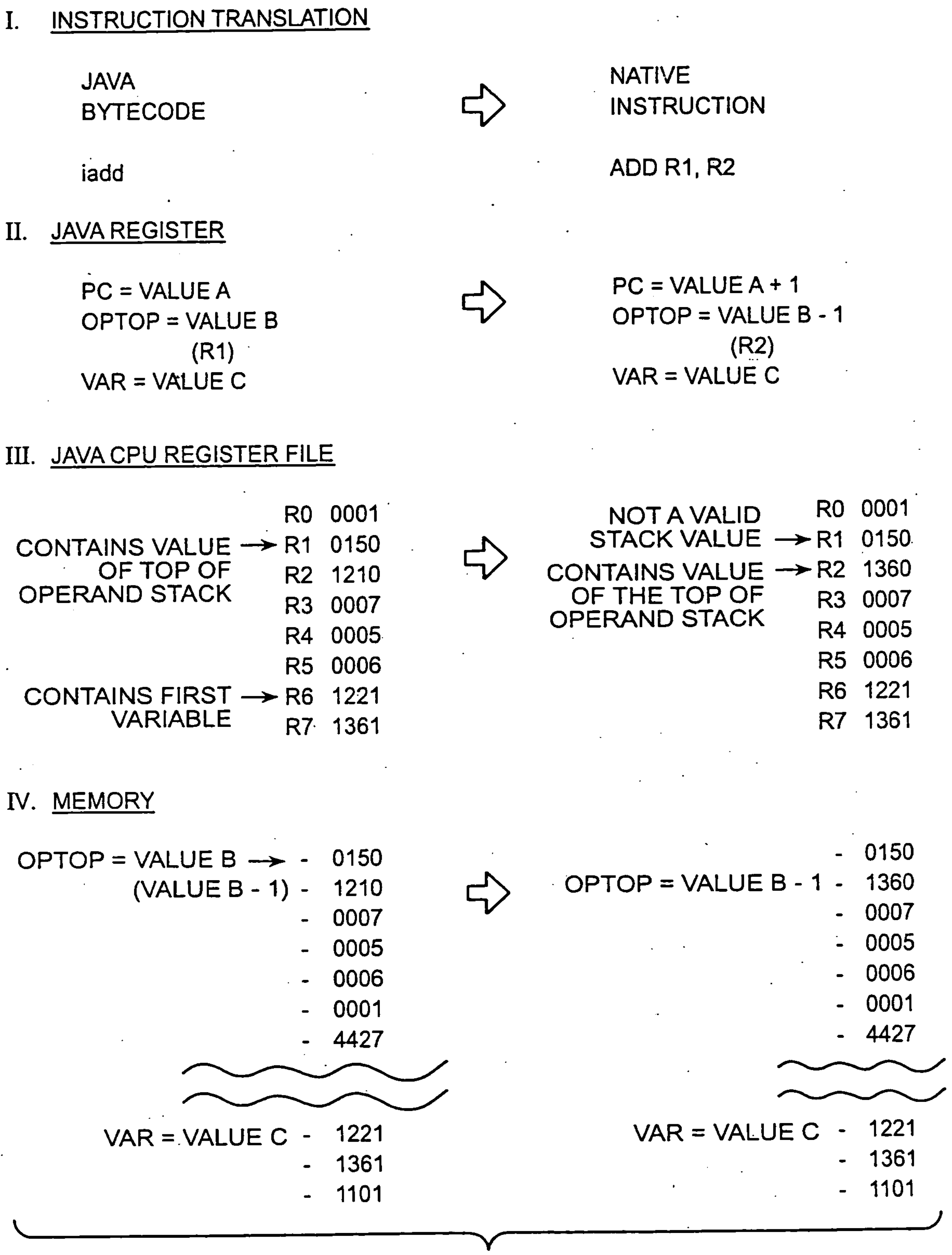

FROM INSTRUCTION CACHE
JAVA ACCELERATOR INSTRUCTION TRANSLATION
42
70
INSTRUCTION BUFFER
72
PARALLEL DECODE
76
LOGIC DATA ALU
74
STATE MACHINE
78
JAVA BYTECODE TO NATIVE INSTRUCTION LOOK-UP TABLE
80
82
BUFFER
NATIVE INSTRUCTIONS
44
JAVA REGISTERS

FIG._4



	Before	After
R0	0001	0001
R1	0150	0150
R2	1210	1360
R3	0007	0007
R4	0005	0005
R5	0006	0006
R6	1221	1221
R7	1361	1361

Before	After
0150	0150
1210	1360
0007	0007
0005	0005
0006	0006
0001	0001
4427	4427
1221	1221
1361	1361
1101	1101

FIG._5

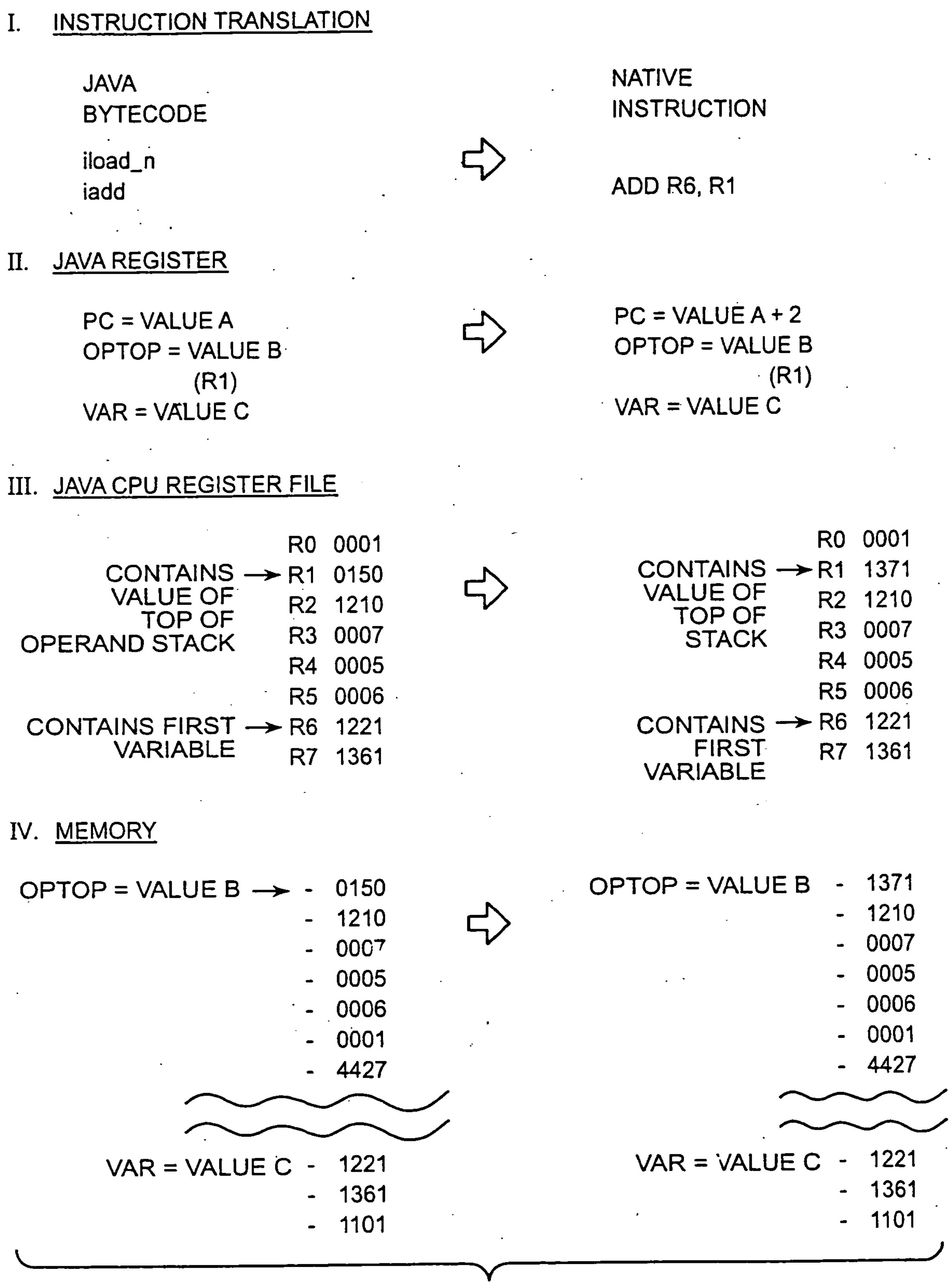

I. INSTRUCTION TRANSLATION
JAVA BYTECODE
iload_n
iadd
NATIVE INSTRUCTION
ADD R6, R1
II. JAVA REGISTER
PC = VALUE A
OPTOP = VALUE B (R1)
VAR = VALUE C
PC = VALUE A + 2
OPTOP = VALUE B (R1)
VAR = VALUE C
III. JAVA CPU REGISTER FILE
CONTAINS VALUE OF TOP OF OPERAND STACK
CONTAINS FIRST VARIABLE
R0 0001
R1 0150
R2 1210
R3 0007
R4 0005
R5 0006
R6 1221
R7 1361
CONTAINS VALUE OF TOP OF STACK
CONTAINS FIRST VARIABLE
R0 0001
R1 1371
R2 1210
R3 0007
R4 0005
R5 0006
R6 1221
R7 1361
IV. MEMORY
OPTOP = VALUE B
- 0150
- 1210
- 0007
- 0005
- 0006
- 0001
- 4427
VAR = VALUE C
- 1221
- 1361
- 1101
OPTOP = VALUE B
- 1371
- 1210
- 0007
- 0005
- 0006
- 0001
- 4427
VAR = VALUE C
- 1221
- 1361
- 1101

FIG._6

Opcodes Mnemonic	Opcode xHH	Excep Gen
nop	0x00	
aconst_null	x01	
iconst_m1	x02	
iconst_n(0-5)	x03 - x08	
lconst_n(0-1)	x09 - x0a	
fconst_n(0-2)	x0c - x0d	
dconst_n(0-1)	x0e -x0f	
bipush	x10	
sipush	x11	
ldc	x12	y
ldc_w	x13	y
ldc2_w	x14	y
iload	x15	
lload	x16	
fload	x17	
dload	x18	
aload	x19	
iload_n(0-3)	x1a - x1d	
lload_n(0-3)	x1e - x21	
fload_n(0-3)	x22 - x25	
dload_n(0-3)	x26 - x29	
aload_n(0-3)	x2a - x2d	
iaload	x2e	
laload	x2f	
faload	x30	
daload	x31	
aaload	x32	
baload	x33	
caload	x34	
saload	x35	
istore	x36	
lstore	x37	
fstore	x38	
dstroe	x39	
astroe	x3a	
istore_n(0-3)	x3b - x3e	
lstore_n(0-3)	x3f - x42	
fstore_n(0-3)	x43 - x46	
dstore_n(0-3)	x47 - x4a	
astore_n(0-3)	x4b - x4e	
iastore	x4f	
lastore	x50	
fastroe	x51	
dastore	x52	
bastore	x53	
aastore	x54	
castroe	x55	
sastore	x56	

FIG._7A

pop	x57	
pop2	x58	
dup	x59	
dup_x1	x5a	
dup_x2	x5b	
dup2	x5c	
dup2_x1	x5d	
dup2_x2	x5e	
swap	x5f	
iadd	x60	
ladd	x61	
fadd	x62	y
dadd	x63	y
isub	x64	
lsub	x65	
fsub	x66	y
dsub	x67	y
imul	x68	
lmul	x69	
fmul	x6a	y
dmul	x6b	y
idiv	x6c	y
ldiv	x6d	y
fdiv	x6e	y
ddiv	x6f	y
irem	x70	y
lrem	x71	y
frem	x72	y
drem	x73	y
ineg	x74	
lneg	x75	
fneg	x76	y
dneg	x77	y
ishl	x78	
lshl	x79	
ishr	x7a	
lshr	x7b	
iushr	x7c	
lushr	x7d	
iand	x7e	
land	x7f	
ior	x80	
lor	x81	
ixor	x82	
lxor	x83	
iinc	x84	
i2l	x85	y
i2f	x86	y
i2d	x87	y
l2i	x88	y
l2f	x89	y
l2d	x8a	y

FIG._7B

f2i	x8b	y
f2l	x8c	y
f2d	x8d	y
d2i	x8e	y
d2l	x8f	y
d2f	x90	y
i2b	x91	
i2c	x92	
i2s	x93	
lcmp	x94	y
fcmpl	x95	y
fcmpg	x96	y
dcmpl	x97	y
dcmpg	x98	y
ifeq	x99	
ifne	x9a	
iflt	x9b	
ifge	x9c	
ifgt	x9d	
ifle	x9e	
if_icmpeq	x9f	
if_icmpne	xa0	
if_icmplt	xa1	
if_acmpge	xa2	
if_cmpgt	xa3	
if_icmple	xa4	
if_acmpeq	xa5	
if_acmpne	xa6	
goto	xa7	
jsr	xa8	
ret	xa9	
tableswitch	xaa	y
lookupswitch	xab	y
ireturn	xac	
lreturn	xad	
freturn	xae	
dreturn	xaf	
areturn	xb0	
return	xb1	
getstatic	xb2	y
putstatic	xb3	y
getfield	xb4	y
putfield	xb5	y
invokevirtual	xb6	y
invokespecial	xb7	y
invokestatic	xb8	y
invokeinterface	xb9	y
xxunsedxxx	xba	y
new	xbb	y
newarray	xbc	y
anewarray	xbd	y
arraylength	xbe	y

FIG._7C

athrow	xbf	y
checkcast	xco	y
instanceof	xc1	y
monitorenter	xc2	y
monitorexit	xc3	y
wide	xc4	y
multianewarray	xc5	y
ifnull	xc6	y
ifnonnull	xc7	y
goto_w	xc8	
jsr_w	xc9	
ldc_quick	xcb	y
ldc_w_quick	xcc	y
ldc2_w_quick	xcd	y
getfield_quick	xce	y
putfield_quick	xcf	y
getfield2_quick	xd0	y
putfield2_quick	xd1	y
getstatic_quick	xd2	y
putstatic_quick	xd3	y
gtestatic2_quick	xd4	y
putstatic2_quick	xd5	y
invokevirtual_quick	xd6	y
invokenonvirtual_quick	xd7	y
invokesuper_quick	xd8	y
invokestatic_quick	xd9	y
invokeinterface_quick	xda	y
invokevirtualobject_quick	xdb	y
new_quick	xdc	y
anewarray_quick	xde	y
multinewarray_quick	xdf	y
checkcast_quick	xe0	y
instanceof_quick	xe1	y
invokevirtual_quick_w	xe2	y
getfield_quick_w	xe3	y
putfield_quick_w	xe4	y
breakpoint	xca	y
impdep1	xfe	y
impdep2	xff	y

FIG._7D

APPLN. FILING DATE: OCTOBER 13, 2000
TITLE: JAVA HARDWARE ACCELERATOR USING MICROCODE ENGINE
INVENTOR(S): PATEL
APPLICATION SERIAL NO: 09/687,777

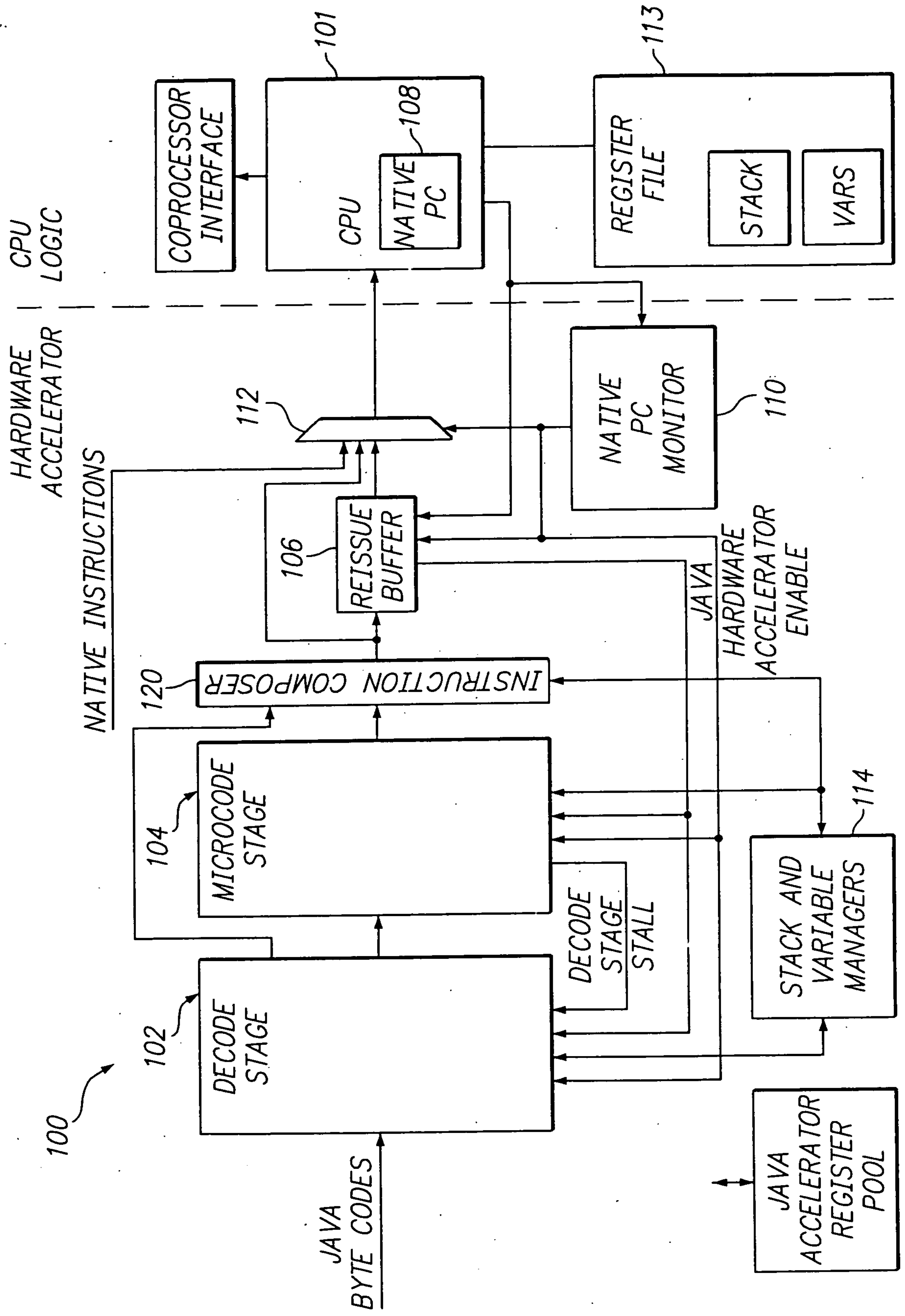


FIG. 8

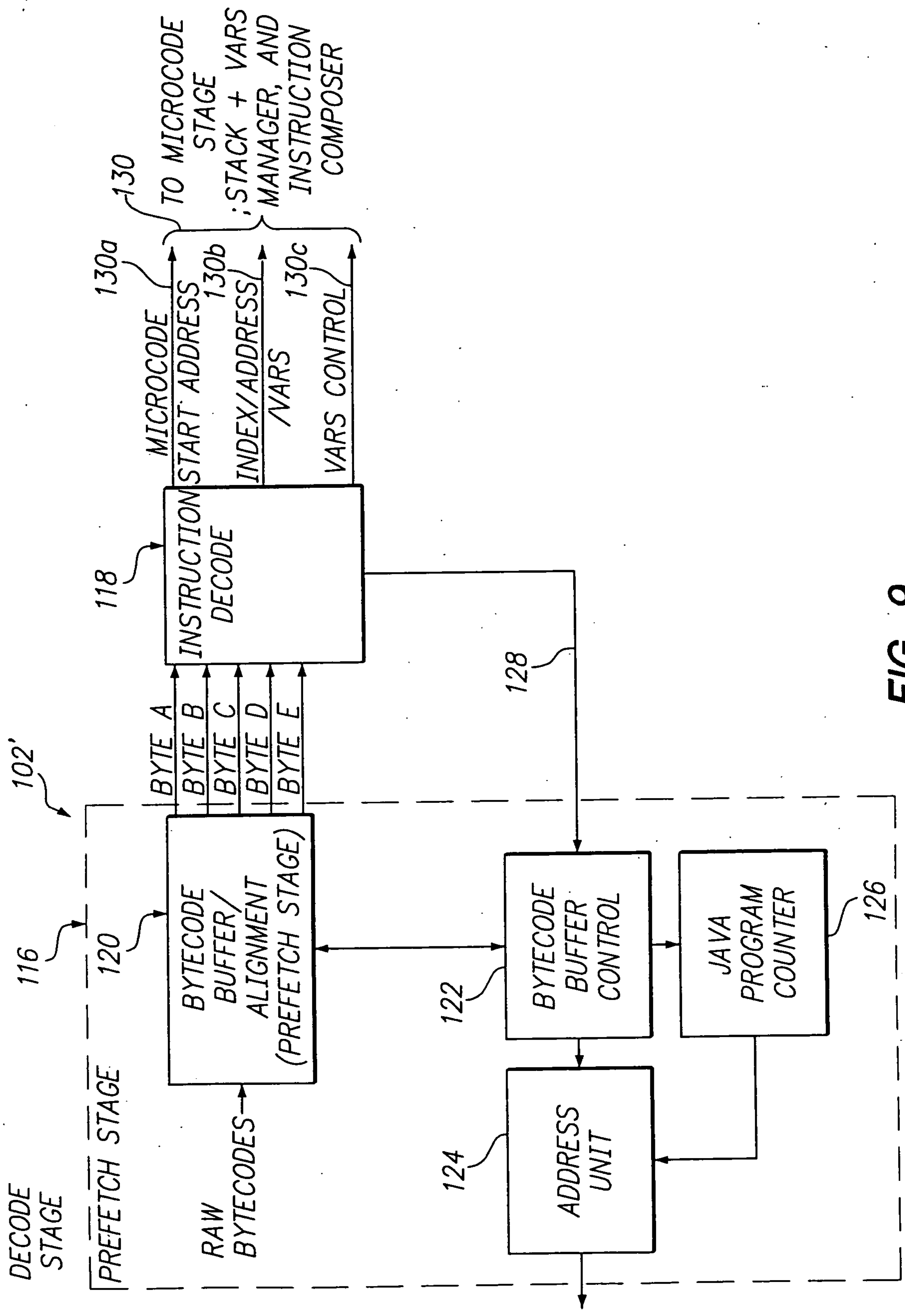

DECODE STAGE
PREFETCH STAGE
116
102'
120
BYTECODE BUFFER/ ALIGNMENT (PREFETCH STAGE)
RAW BYTECODES
BYTE A
BYTE B
BYTE C
BYTE D
BYTE E
118
INSTRUCTION DECODE
MICROCODE START ADDRESS
130a
INDEX/ADDRESS/ /VARS
130b
VARS CONTROL
130c
130
TO MICROCODE STAGE ;STACK + VARS MANAGER, AND INSTRUCTION COMPOSER
128
122
BYTECODE BUFFER CONTROL
126
JAVA PROGRAM COUNTER
124
ADDRESS UNIT

FIG. 9

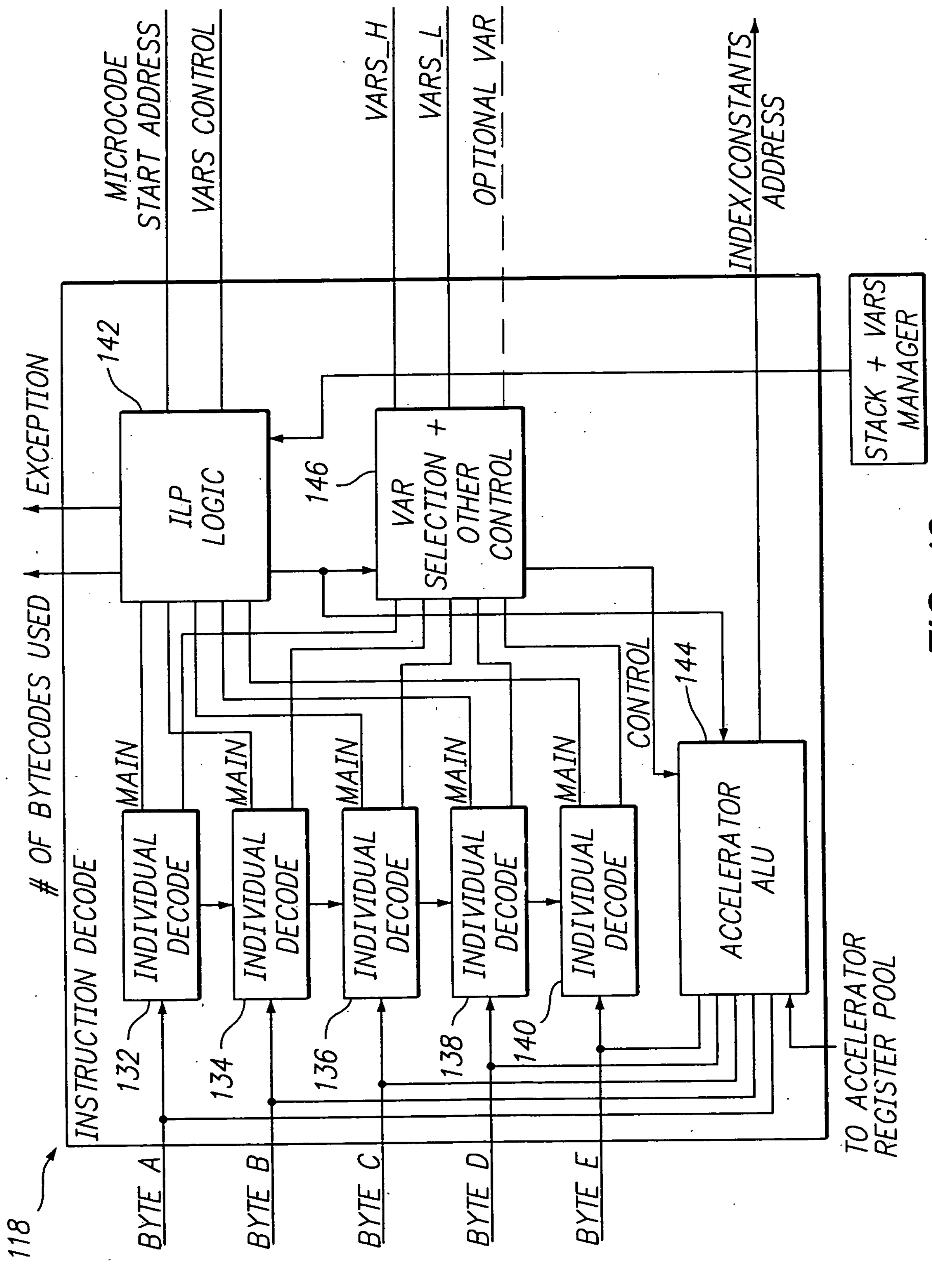

118
INSTRUCTION DECODE
# OF BYTECODES USED
EXCEPTION
MICROCODE START ADDRESS
VARS CONTROL
VARS_H
VARS_L
OPTIONAL_VAR
INDEX/CONSTANTS ADDRESS
142
ILP LOGIC
146
VAR SELECTION + OTHER CONTROL
STACK + VARS MANAGER
CONTROL
144
ACCELERATOR ALU
MAIN
MAIN
MAIN
MAIN
MAIN
INDIVIDUAL DECODE
INDIVIDUAL DECODE
INDIVIDUAL DECODE
INDIVIDUAL DECODE
INDIVIDUAL DECODE
132
134
136
138
140
BYTE A
BYTE B
BYTE C
BYTE D
BYTE E
TO ACCELERATOR REGISTER POOL

FIG. 10

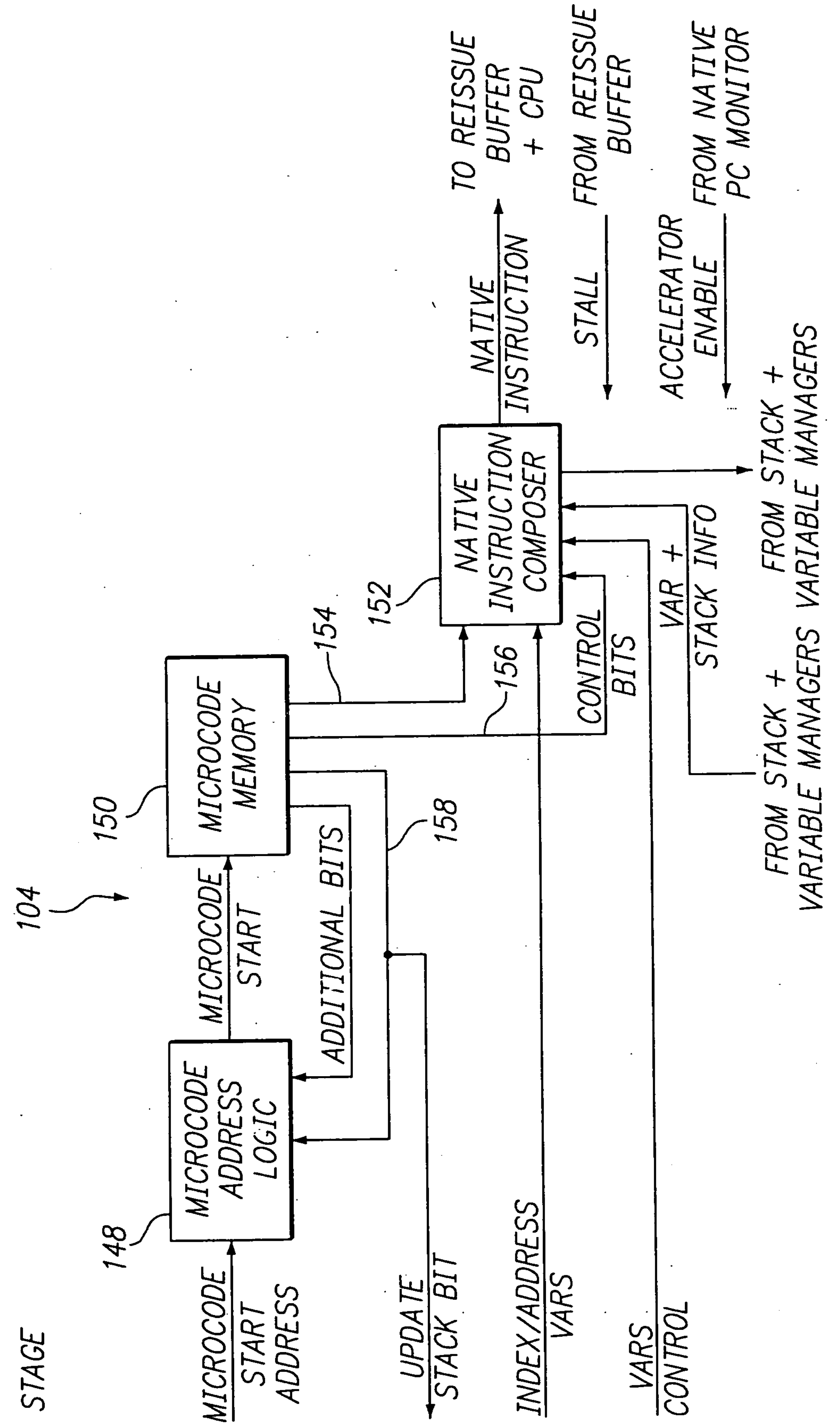

104
STAGE
MICROCODE START ADDRESS
148
MICROCODE ADDRESS LOGIC
MICROCODE START
ADDITIONAL BITS
150
MICROCODE MEMORY
158
154
156
CONTROL BITS
152
NATIVE INSTRUCTION COMPOSER
NATIVE INSTRUCTION
TO REISSUE BUFFER + CPU
STALL
FROM REISSUE BUFFER
ACCELERATOR ENABLE
FROM NATIVE PC MONITOR
FROM STACK + VARIABLE MANAGERS
VAR + STACK INFO
FROM STACK + VARIABLE MANAGERS
UPDATE STACK BIT
INDEX/ADDRESS VARS
VARS CONTROL

FIG. 11

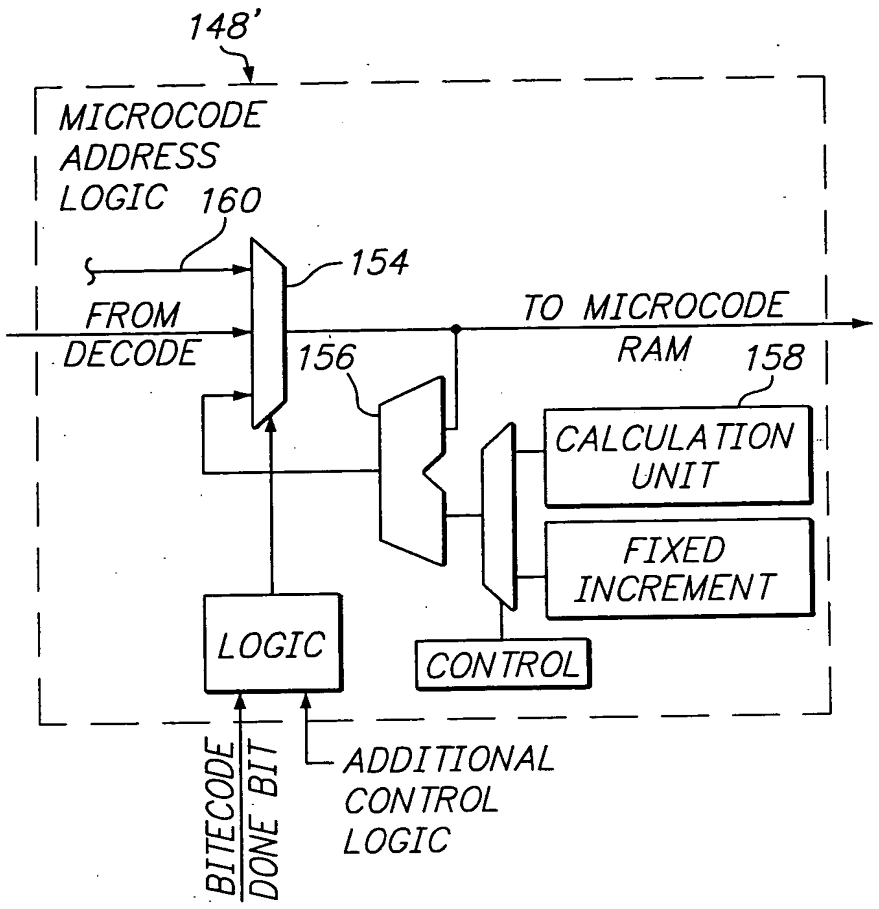

148'
MICROCODE ADDRESS LOGIC
160
154
FROM DECODE
TO MICROCODE RAM
156
158
CALCULATION UNIT
FIXED INCREMENT
LOGIC
CONTROL
BITECODE
DONE BIT
ADDITIONAL CONTROL LOGIC

FIG. 12

APPLN. FILING DATE: OCTOBER 13, 2000
TITLE: JAVA HARDWARE ACCELERATOR USING MICROCODE ENGINE
INVENTOR(S): PATEL
APPLICATION SERIAL NO: 09/687,777 **SHEET 15 of 19**

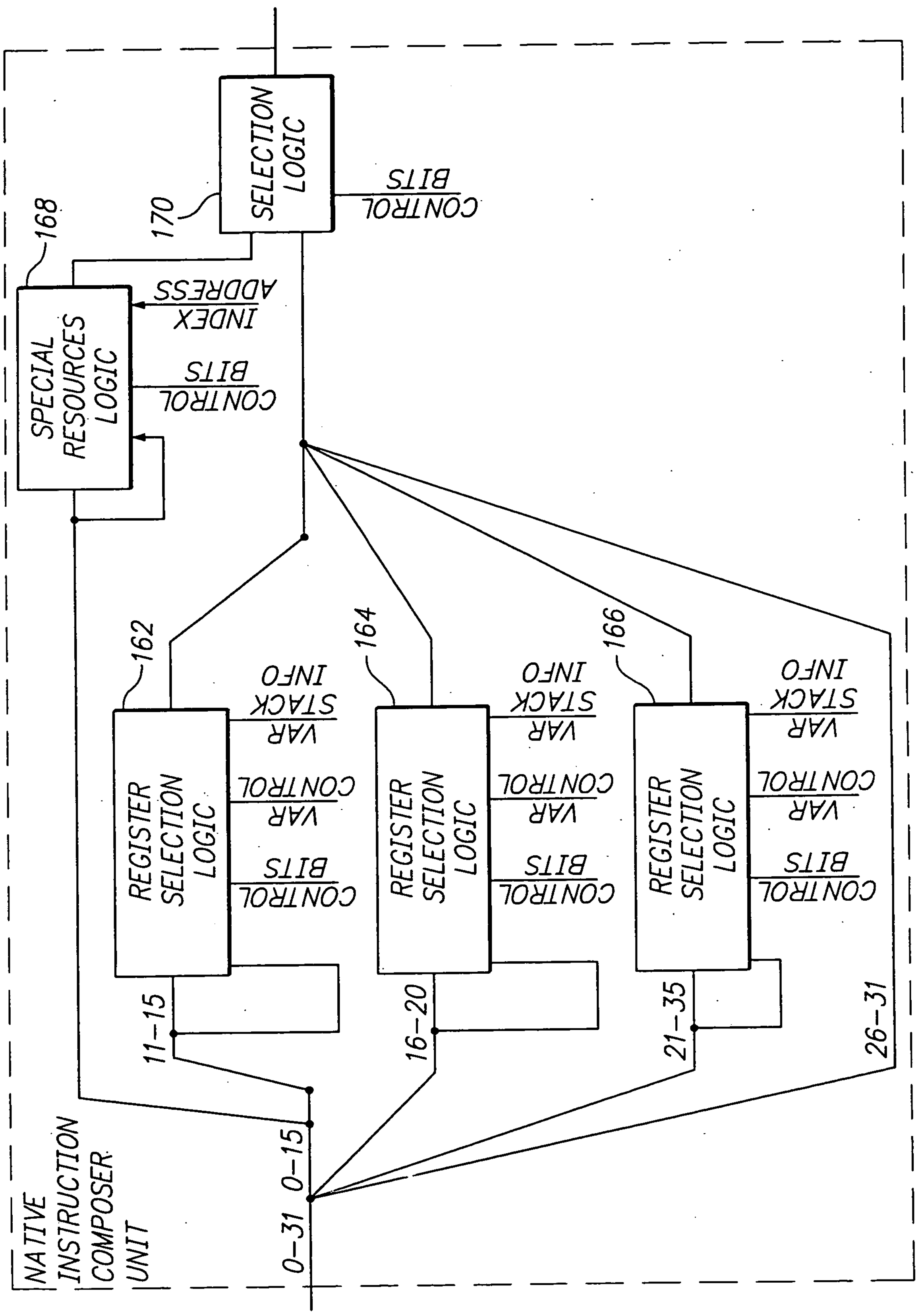


FIG. 13

APPLN. FILING DATE: OCTOBER 13, 2000
TITLE: JAVA HARDWARE ACCELERATOR USING MICROC[illegible]DE ENGINE
INVENTO[illegible]): PATEL
APPLICATION SERIAL NO: 09/687,777

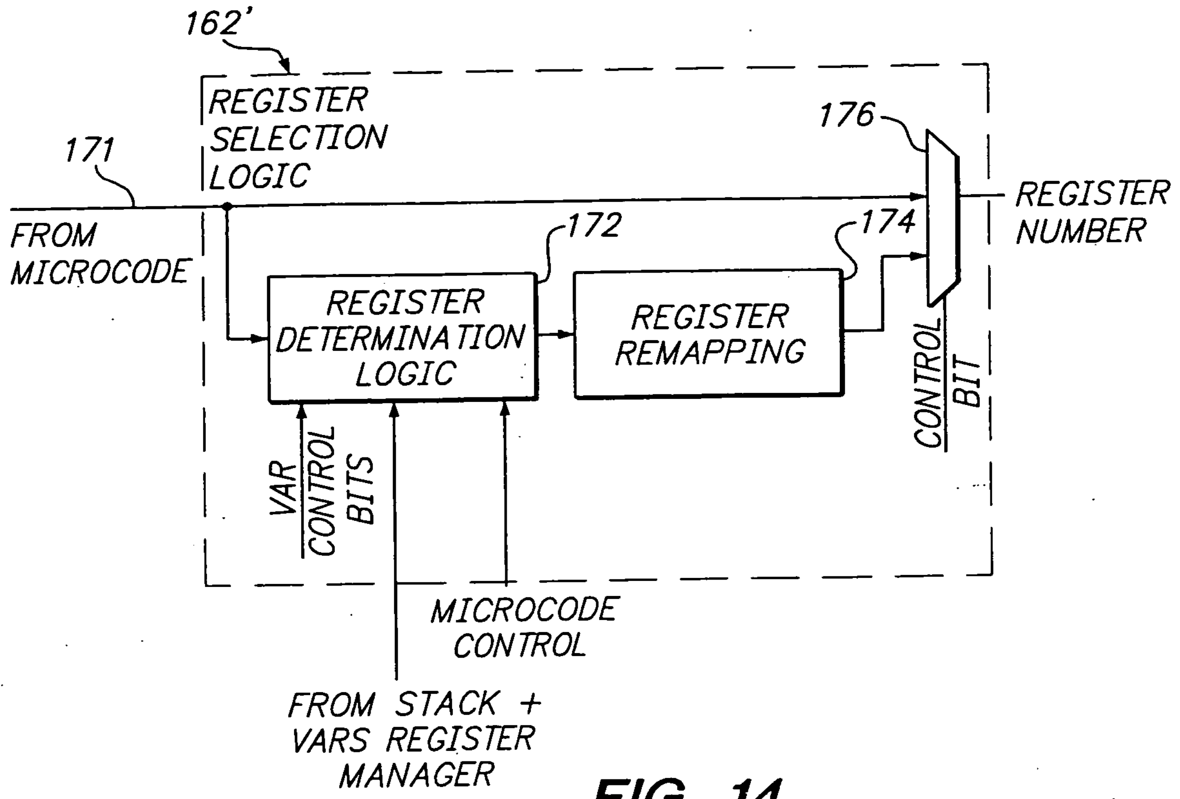


FIG. 14

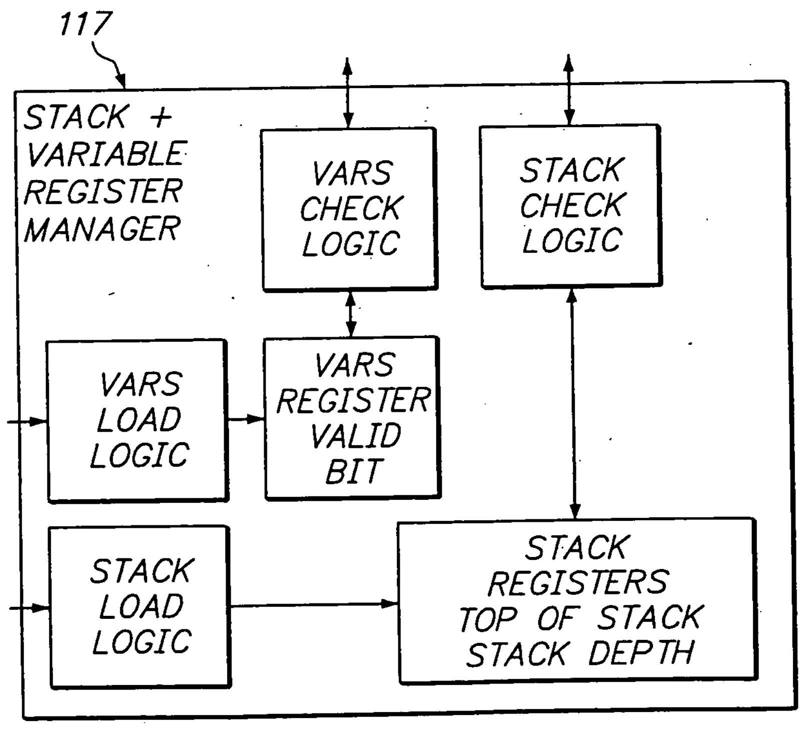


FIG. 15

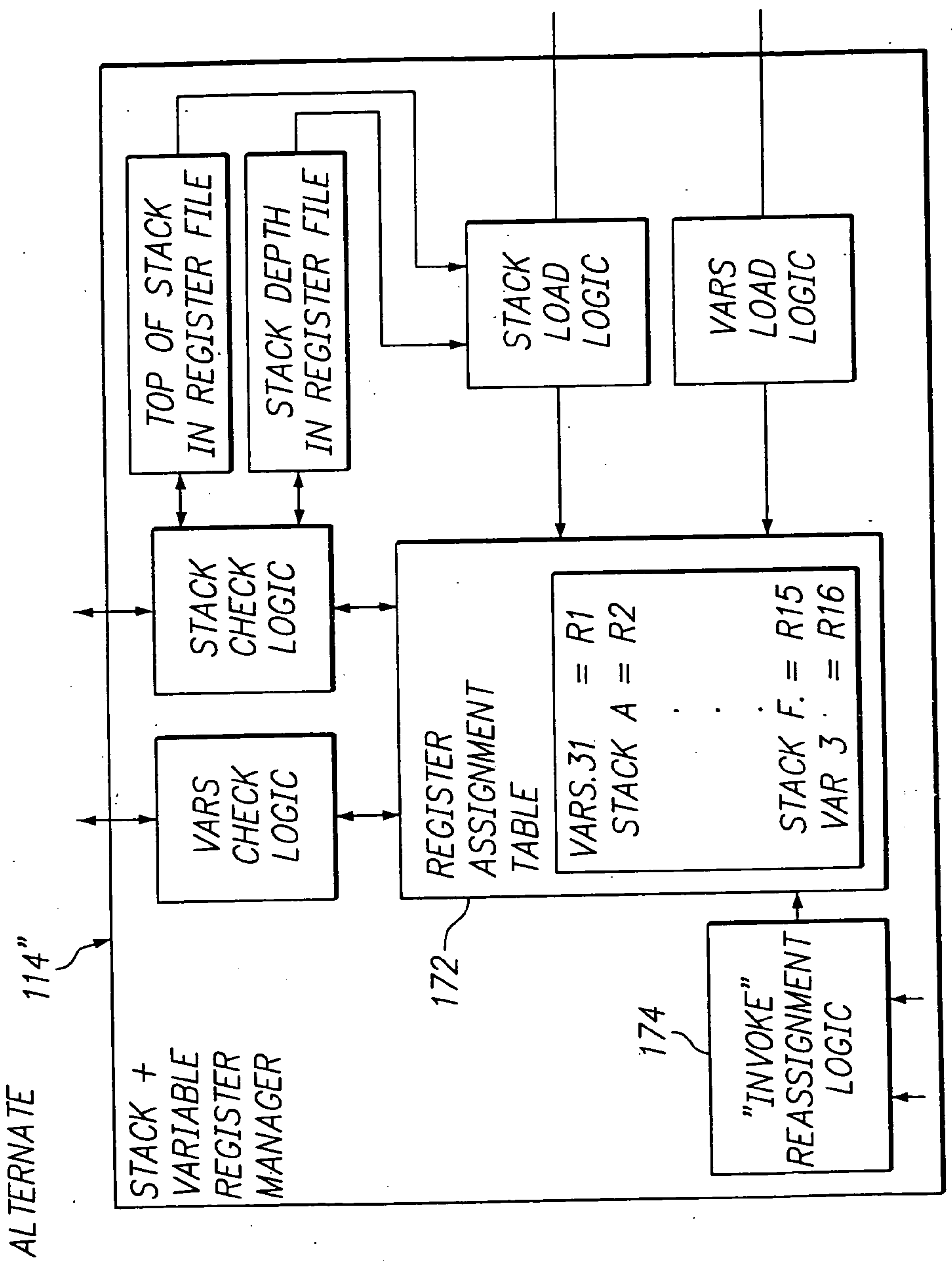

ALTERNATE
114"
STACK + VARIABLE REGISTER MANAGER
TOP OF STACK IN REGISTER FILE
STACK DEPTH IN REGISTER FILE
STACK LOAD LOGIC
VARS LOAD LOGIC
STACK CHECK LOGIC
VARS CHECK LOGIC
REGISTER ASSIGNMENT TABLE
VARS.31 = R1
STACK A = R2
STACK F. = R15
VAR 3 = R16
172
174
"INVOKE" REASSIGNMENT LOGIC

FIG. 16

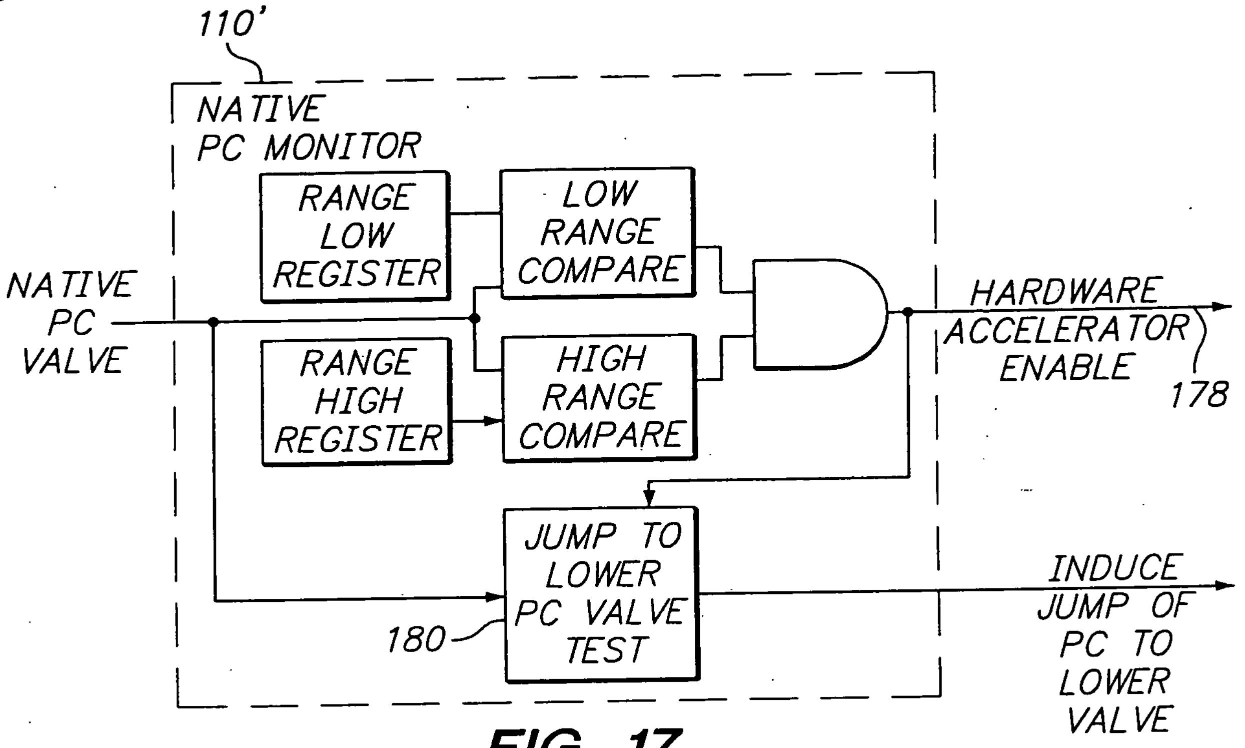

110'
NATIVE PC MONITOR
RANGE LOW REGISTER
LOW RANGE COMPARE
RANGE HIGH REGISTER
HIGH RANGE COMPARE
NATIVE PC VALVE
HARDWARE ACCELERATOR ENABLE
178
JUMP TO LOWER PC VALVE TEST
180
INDUCE JUMP OF PC TO LOWER VALVE

FIG. 17

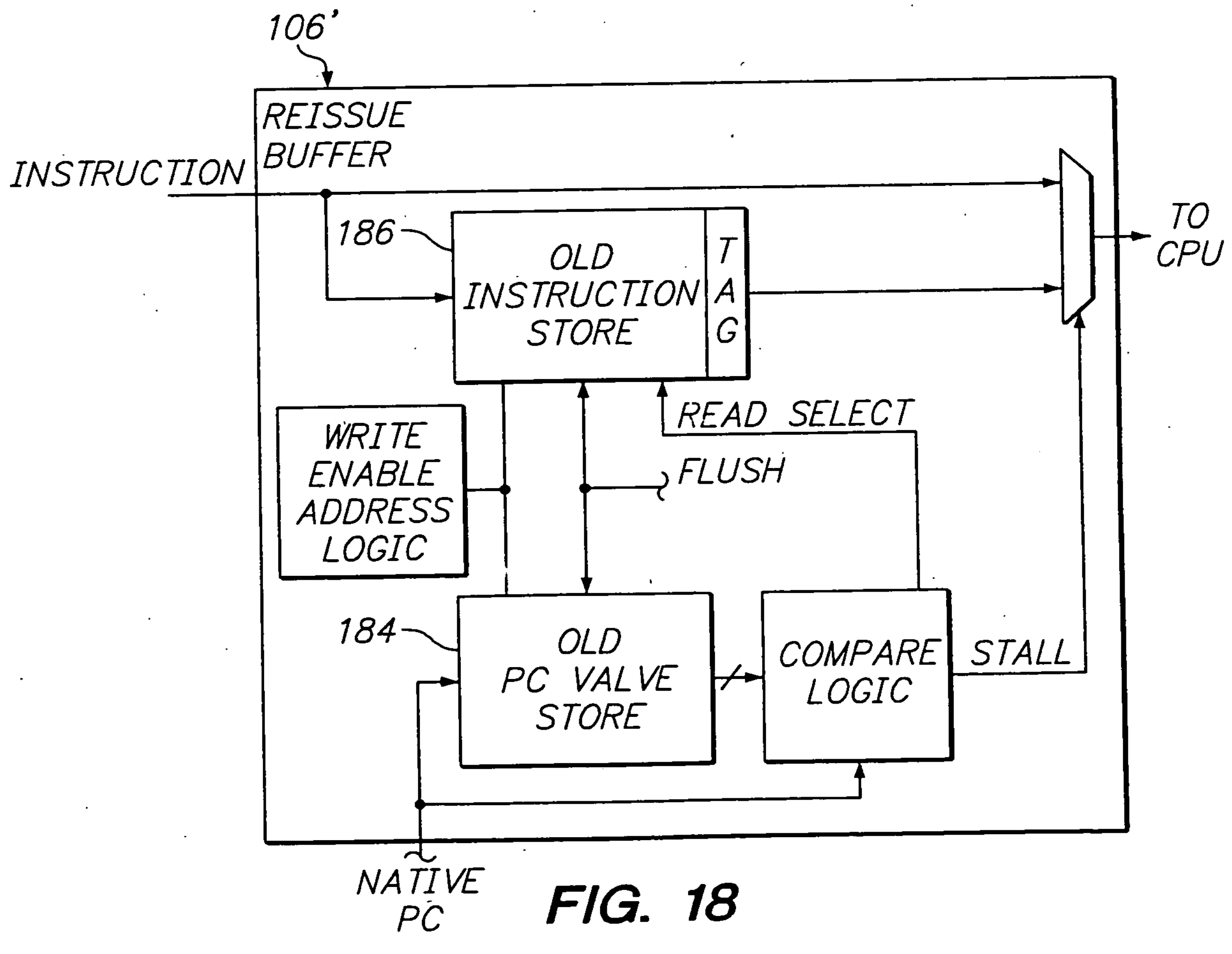
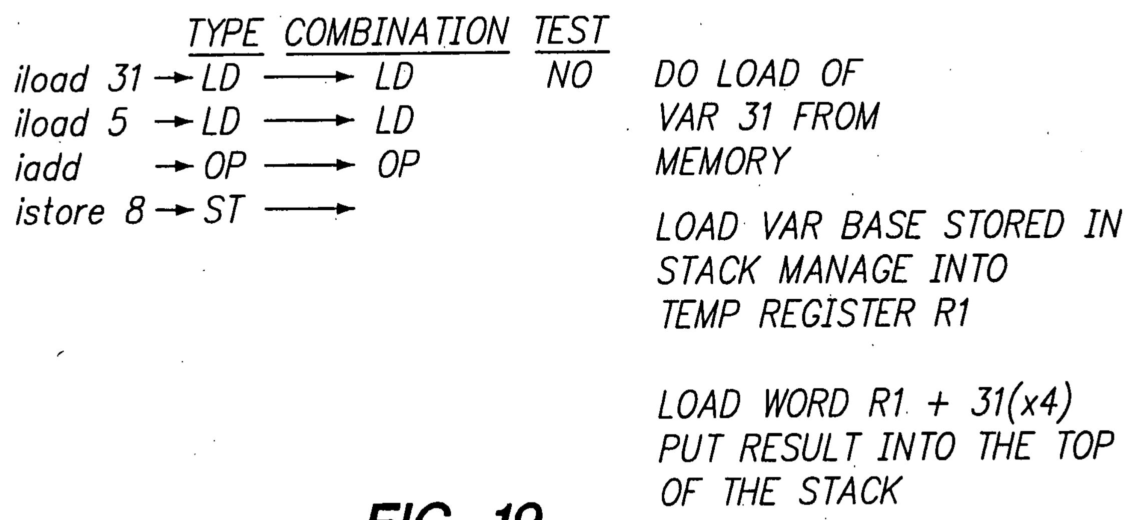

106'
REISSUE BUFFER
INSTRUCTION
186
OLD INSTRUCTION STORE
TAG
TO CPU
WRITE ENABLE ADDRESS LOGIC
READ SELECT
FLUSH
184
OLD PC VALVE STORE
COMPARE LOGIC
STALL
NATIVE PC

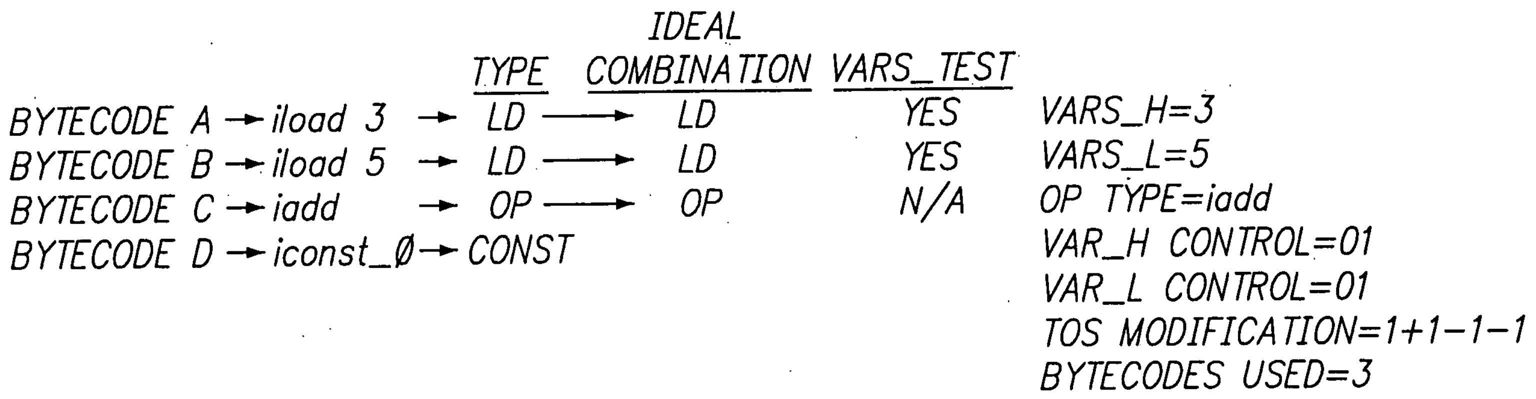
FIG. 18



```
            TYPE  COMBINATION  TEST
iload 31 → LD  ———→  LD        NO     DO LOAD OF
iload 5  → LD  ———→  LD               VAR 31 FROM
iadd     → OP  ———→  OP               MEMORY
istore 8 → ST  ———→
                                      LOAD VAR BASE STORED IN
                                      STACK MANAGE INTO
                                      TEMP REGISTER R1

                                      LOAD WORD R1 + 31(x4)
                                      PUT RESULT INTO THE TOP
                                      OF THE STACK
```

FIG. 19



```
                                     IDEAL
                           TYPE   COMBINATION  VARS_TEST
BYTECODE A → iload 3    →  LD  ———→  LD          YES      VARS_H=3
BYTECODE B → iload 5    →  LD  ———→  LD          YES      VARS_L=5
BYTECODE C → iadd       →  OP  ———→  OP          N/A      OP TYPE=iadd
BYTECODE D → iconst_Ø → CONST                             VAR_H CONTROL=01
                                                          VAR_L CONTROL=01
                                                          TOS MODIFICATION=1+1−1−1
                                                          BYTECODES USED=3
```

FIG. 20